6502 revival

The rapidly changing world of technology means that processors quickly become anachronistic. One such example is the 6502. Although still a satisfactory performer, it looks a bit jaded in the light of recent developments. Simon Taylor and Bob Watford explain how Rockwell International has given it a 'facelift'.

The 6502 microprocessor has been with us for a good few years now; reputable computers using the 6502 include the first 'Personal Computers', the PET and the Apple II, and more recently the Oric, VIC-20 and the BBC Micro.

This fine old workhorse sometimes looks a little jaded compared to some of the more recent developments such as the 68000 and even the 6809 and could do with something of a 'facelift'.

Rockwell International, one of the manufacturers of the 6502, has recognised the need for a 6502 facelift and produced a new, pin-compatible, instruction-compatible 6502 processor.

The R65C02, as it is called, is manufactured in CMOS technology which gives it some distinct advantages over NMOS:

(1) Power consumption at 1MHz is only 20mw compared with the NMOS consumption of 450mw. This will also allow battery-run computers to be designed using the new CMOS peripheral chips.

(2) CMOS also has a far better tolerance to supply voltage and, of course, better noise immunity.

(3) CMOS, due to its lower power consumption and heat dissipation, makes possible a more compact architecture, which, in turn, has allowed Rockwell to add more instructions to the CPU.

(courtesy Rockwell International).

Hex	Mnemonic	Addressing mode	Description
72	ADC	ZP Indirect (1)	Add memory to accumulator with carry
32	AND	ZP Indirect (1)	AND memory with accumulator
OF-7F	BBR (2)	Zero page	Branch on bit reset
8F-FF	BBS(2)	Zero Page	Branch on bit set
3C	BIT	Absolute,X	Test memory bits with accumulator
34	BIT	Zero page,X	
89	BIT	Immediate	
80	BRA	Relative	Branch relative always
D2	CMP	ZP Indirect (1)	Compare memory with accumulator
3A	DEC	Accumulator	Decrement accumulator
52	EOR	ZP Indirect (1)	EXOR memory with accumulator
1A	INC	Accumulator	Increment accumulator
7C	JMP	Indirect,X (1)	Jump
B2	LDA	ZP Indirect (1)	Load accumulator with memory
12	ORA	ZP Indirect (1)	OR memory with accumulator
DA	PHX	Implied	Push X on stack
5A	PHY	Implied	Push Y on stack
FA	PLX	Implied	Pull X from stack
7A	PLY	Implied	Pull Y from stack
07-77	RMB (2)	Zero page	Reset memory bit
F2	SBC	ZP Indirect	Subtract memory from accumulator with borrow
87-F7	SMB (2)	Zero page	Set memory bit
12	STA	ZP Indirect	Store accumulator in memory
9C	STZ	Absolute	Store zero
9E	STZ	Absolute,X	
64	STZ	Zero page	
74	STZ	Zero page,X	
1C	TRB	Absolute	Test and reset memory bits with accumulator
14	TRB	Zero page	
0C	TSB	Absolute	Test and set memory bits with accumulator
04	TSB	Zero page	

Fig 1 Opcodes

Using an R65C02

Let's look at these extra instructions and various methods of utilising an R65C02 in an existing micro (Fig 1).

Zero page indirect and indirect indexed by X are totally new addressing modes. In the zero page indirect mode, the second byte contains a zero page address containing the 16-bit address that gives the effective address. Indirect indexed by X takes the second and third bytes of the instruction and adds these to the X register to give the effective address. This is useful for implementations of jump tables.

The most significant digit will change in the opcode to give the bit being tested: that is, the opcode for SMB 2 is A7.

Many 6502 programs will contain sequences such as:

```
TXA
PHA
TYA
PHA
```

to save the index registers on the stack, or

```
LDA $00
ORA £$01
STA $00
```

to set a bit in a particular byte in zerc page.

The above examples use four bytes and 10 clock cycles, and six bytes and eight clock cycles respectively.

With the 65C02, however, these become:

```
PHX
PHY
```

and

```
SMB 0,$00
```

These are equivalent program sequences performing exactly the same function, but they do not destroy the accumulator. They also use only two bytes each and six and five clock cycles

respectively. Hence, on most operations, savings of memory space and program execution time can be achieved.

The main problem, however, when writing software for this processor is that the new opcodes are not included in the existing assembler! This can be solved by using .BYTE operands for all but the branch instructions, or, in the case of the BBC Micro, new functions can be defined for the new opcodes and the OPT directive can be used.

Implementation

To fit an R65C02 in your computer, you must first check the machine's running speed.

'Rockwell International, one of the manufacturers of the 6502, has recognised the need for a 6502 facelift and produced a new, pin-compatible, instruction-compatible 6502 processor.'

The following table will tell you what type of R65C02 to obtain for the relevant speeds:

1MHz R65C02P1
2MHz R65C02P2
3MHz R65C02P3
4MHz R65C02P4

So, for a BBC Micro you will need an R65C02P2.

1- and 2MHz versions are available now, and 3- and 4MHz versions will be available later this year.

Rockwell distributor, RCS Microsystems has put together a pack for BBC Micro owners consisting of an R65C02P2, a listing of a program enabling the new opcodes to be used in the BBC assembler (Fig 2) and an R65C02 data sheet.

The price is £17.25, incl VAT & p&p. Further information is available on (01) 979 2204. END

```
!ST
 10 REM ADDING 65C02 OP-CODES TO THE
       BBC ASSEMBLER
 20 REM PROGRAM BY BOB WATFORD
 30 REM
 40 REM THE FIRST PART OF THE PROGRAM
       SHOWS HOW THE NEW OP-CODES
       ARE USED
 50 REM
 60 REM LINES BELOW 30000 MAY BE
       DELETED, AND REPLACED BY YOUR
       OWN CODE
 70 REM
 80 REM Z% SHOULD BE REPLACED BY THE
       VARIABLE USED FOR 'OPT'
       (eg. opt, PASS ETC)
 90 REM
100DIM CODE% &100
110TEST=&12:BGTEST=&ABCD
120FORZ%=4 TO 7 STEP 3
130O%=CODE%:P%=&1000
140[
150OPT Z%
160OPT FNBRA(Z%,LABEL1)
170OPT FNORA(Z%,TEST)
180OPT FNAND(Z%,TEST)
190OPT FNEOR(Z%,TEST)
200OPT FNADC(Z%,TEST)
210OPT FNSTA(Z%,TEST)
220OPT FNLDA(Z%,TEST)
230OPT FNCMP(Z%,TEST)
240OPT FNSBC(Z%,TEST)
250.LABEL1 OPT FNTSB(Z%,TEST)
260OPT FNTRB(Z%,TEST)
270OPT FNBIT(Z%,TEST)
280OPT FNSTZ(Z%,TEST)
290OPT FNSTZX(Z%,TEST)
300OPT FNRMB(Z%,0,TEST)
310OPT FNRMB(Z%,1,TEST)
320OPT FNRMB(Z%,2,TEST)
330OPT FNRMB(Z%,3,TEST)
340OPT FNRMB(Z%,4,TEST)
350OPT FNRMB(Z%,5,TEST)
360OPT FNRMB(Z%,6,TEST)
370OPT FNRMB(Z%,7,TEST)
380OPT FNSMB(Z%,0,TEST)
390OPT FNSMB(Z%,1,TEST)
400OPT FNSMB(Z%,2,TEST)
410OPT FNSMB(Z%,3,TEST)
420OPT FNSMB(Z%,4,TEST)
430OPT FNSMB(Z%,5,TEST)
440OPT FNSMB(Z%,6,TEST)
450OPT FNSMB(Z%,7,TEST)
460OPT FNBITI(Z%,&AA)
470OPT FNINCA(Z%)
480OPT FNDECA(Z%)
490OPT FNPHY(Z%)
500OPT FNPLY(Z%)
510OPT FNPHX(Z%)
520OPT FNPLX(Z%)
530OPT FNTSB(Z%,BGTEST)
540OPT FNTRB(Z%,BGTEST)
550OPT FNBIT(Z%,BGTEST)
560OPT FNJMP(Z%,BGTEST)
570OPT FNSTZ(Z%,BGTEST)
580OPT FNSTZX(Z%,BGTEST)
590OPT FNBBR(Z%,0,TEST,L2)
600.L2 OPT FNBBR(Z%,1,TEST,L3)
610.L3 OPT FNBBR(Z%,2,TEST,L4)
620.L4 OPT FNBBR(Z%,3,TEST,L5)
630.L5 OPT FNBBR(Z%,4,TEST,L6)
640.L6 OPT FNBBR(Z%,5,TEST,L7)
650.L7 OPT FNBBR(Z%,6,TEST,L8)
660.L8 OPT FNBBR(Z%,7,TEST,L9)
670.L9 OPT FNBBS(Z%,0,TEST,L10)
680.L10 OPT FNBBS(Z%,1,TEST,L11)
690.L11 OPT FNBBS(Z%,2,TEST,L12)
700.L12 OPT FNBBS(Z%,3,TEST,L13)
710.L13 OPT FNBBS(Z%,4,TEST,L14)
720.L14 OPT FNBBS(Z%,5,TEST,L15)
730.L15 OPT FNBBS(Z%,6,TEST,L16)
740.L16 OPT FNBBS(Z%,7,TEST,L9)
750]
760NEXT
770 REM THIS IS THE END OF
       THE TEST PROGRAM
30000REM  START OF 65C02 FUNCTIONS
30001END
30002DEFFNBRA(opt%,where%)
30003[OPT opt%:BNE where%:]
30004IFopt%>3 O%?-2=&80 ELSEP%?-2=&80
30005=opt%
30006DEFFNBBS(opt%,bit%,address%,where%)
30007[OPT opt%:BNE where%:]
30008IFopt%>3 O%?-2=&8F+bit%*16:O%?0=(O%
?-1)-1:O%?-1=address%:O%=O%+1 ELSEP%?-2=
&8F+bit%*16:P%?0=(P%?-1)-1:P%?-1=address
%
30009P%=P%+1
30010=opt%
30011DEFFNBBR(opt%,bit%,address%,where%)
30012[OPT opt%:BNE where%:]
30013IFopt%>3 O%?-2=&0F+bit%*16:O%?0=(O%
?-1)-1:O%?-1=address%:O%=O%+1 ELSEP%?-2=
&0F+bit%*16:P%?0=(P%?-1)-1:P%?-1=address
%
30014P%=P%+1
30015=opt%
30016DEFFNPHY(opt%)
30017[OPT opt%:EQUB &5A:]
30018=opt%
30019DEFFNPLY(opt%)
30020[OPT opt%:EQUB &7A:]
30021=opt%
30022DEFFNPHX(opt%)
30023[OPT opt%:EQUB &DA:]
30024=opt%
30025DEFFNPLX(opt%)
30026[OPT opt%:EQUB &FA:]
30027=opt%
30028DEFFNRMB(opt%,bit%,address%)
30029[OPT opt%:EQUB bit%*16+&07:EQUB add
ress%:]
30030=opt%
30031DEFFNSMB(opt%,bit%,address%)
30032[OPT opt%:EQUB bit%*16+&87:EQUB add
ress%:]
30033=opt%
30034DEFFNINCA(opt%)
30035[OPT opt%:EQUB &1A:]
30036=opt%
30037DEFFNDECA(opt%)
30038[OPT opt%:EQUB &3A:]
30039=opt%
30040DEFFNTSB(opt%,address%)
30041IFaddress%>&FF [OPT opt%:EQUB &0C:E
QUW address%:] ELSE[OPT opt%:EQUB &04:EQ
UB address%:]
30042=opt%
30043DEFFNTRB(opt%,address%)
30044IFaddress%>&FF [OPT opt%:EQUB &1C:E
QUW address%:] ELSE[OPT opt%:EQUB &14:EQ
UB address%:]
30045=opt%
30046DEFFNSTZ(opt%,address%)
30047IFaddress%>&FF [OPT opt%:EQUB &9C:E
QUW address%:] ELSE[OPT opt%:EQUB &64:EQ
UB address%:]
30048=opt%
30049DEFFNSTZX(opt%,address%)
30050IFaddress%>&FF [OPT opt%:EQUB &9E:E
QUW address%:] ELSE[OPT opt%:EQUB &74:EQ
UB address%:]
30051=opt%
30052DEFFNORA(opt%,address%)
30053[OPT opt%:EQUB &12:EQUB address%:]
30054=opt%
30055DEFFNAND(opt%,address%)
30056[OPT opt%:EQUB &32:EQUB address%:]
30057=opt%
30058DEFFNEOR(opt%,address%)
30059[OPT opt%:EQUB &52:EQUB address%:]
30060=opt%
30061DEFFNADC(opt%,address%)
30062[OPT opt%:EQUB &72:EQUB address%:]
30063=opt%
30064DEFFNSTA(opt%,address%)
30065[OPT opt%:EQUB &92:EQUB address%:]
30066=opt%
30067DEFFNLDA(opt%,address%)
30068[OPT opt%:EQUB &B2:EQUB address%:]
30069=opt%
30070DEFFNCMP(opt%,address%)
30071[OPT opt%:EQUB &D2:EQUB address%:]
30072=opt%
30073DEFFNSBC(opt%,address%)
30074[OPT opt%:EQUB &F2:EQUB address%:]
30075=opt%
30076DEFFNJMP(opt%,address%)
30077[OPT opt%:EQUB &7C:EQUW address%:]
30078=opt%
30079DEFFNBIT(opt%,address%)
30080IFaddress%>&FF [OPT opt%:EQUB &3C:E
QUW address%:] ELSE[OPT opt%:EQUB &34:EQ
UB address%:]
30081=opt%
30082DEFFNBITI(opt%,byte%)
30083[OPT opt%:EQUB &89:EQUB byte%:]
30084=opt%
```

Fig 2 Program listing

(courtesy Rockwell International).

Hex	Mnemonic	Addressing mode	Description
72	ADC	ZP Indirect (1)	Add memory to accumulator with carry
32	AND	ZP Indirect (1)	AND memory with accumulator
OF-7F	BBR (2)	Zero page	Branch on bit reset
8F-FF	BBS(2)	Zero Page	Branch on bit set
3C	BIT	Absolute,X	Test memory bits with accumulator
34	BIT	Zero page,X	
89	BIT	Immediate	
80	BRA	Relative	Branch relative always
D2	CMP	ZP Indirect (1)	Compare memory with accumulator
3A	DEC	Accumulator	Decrement accumulator
52	EOR	ZP Indirect (1)	EXOR memory with accumulator
1A	INC	Accumulator	Increment accumulator
7C	JMP	Indirect,X (1)	Jump
B2	LDA	ZP Indirect (1)	Load accumulator with memory
12	ORA	ZP Indirect (1)	OR memory with accumulator
DA	PHX	Implied	Push X on stack
5A	PHY	Implied	Push Y on stack
FA	PLX	Implied	Pull X from stack
7A	PLY	Implied	Pull Y from stack
07-77	RMB (2)	Zero page	Reset memory bit
F2	SBC	ZP Indirect	Subtract memory from accumulator with borrow
87-F7	SMB (2)	Zero page	Set memory bit
12	STA	ZP Indirect	Store accumulator in memory
9C	STZ	Absolute	Store zero
9E	STZ	Absolute,X	
64	STZ	Zero page	
74	STZ	Zero page,X	
1C	TRB	Absolute	Test and reset memory bits with accumulator
14	TRB	Zero page	
0C	TSB	Absolute	Test and set memory bits with accumulator
04	TSB	Zero page	

Fig 1 Opcodes

```
IST
 10 REM ADDING 65C02 OP-CODES TO THE
       BBC ASSEMBLER
 20 REM PROGRAM BY BOB WATFORD
 30 REM
 40 REM THE FIRST PART OF THE PROGRAM
       SHOWS HOW THE NEW OP-CODES
       ARE USED

 50 REM
 60 REM LINES BELOW 30000 MAY BE
       DELETED, AND REPLACED BY YOUR
       OWN CODE

 70 REM
 80 REM Z% SHOULD BE REPLACED BY THE
       VARIABLE USED FOR 'OPT'
       (eg. opt, PASS ETC)

 90 REM
100DIM CODE% &100
110TEST=&12:BGTEST=&ABCD
120FORZ%=4 TO 7 STEP 3
130P%=CODE%:P%=&1000
140[
150OPT Z%
160OPT FNBRA(Z%,LABEL1)
170OPT FNORA(Z%,TEST)
180OPT FNAND(Z%,TEST)
190OPT FNEOR(Z%,TEST)
200OPT FNADC(Z%,TEST)
210OPT FNSTA(Z%,TEST)
220OPT FNLDA(Z%,TEST)
230OPT FNCMP(Z%,TEST)
240OPT FNSBC(Z%,TEST)
250.LABEL1 OPT FNTSB(Z%,TEST)
260OPT FNTRB(Z%,TEST)
270OPT FNBIT(Z%,TEST)
280OPT FNSTZ(Z%,TEST)
290OPT FNSTZX(Z%,TEST)
300OPT FNRMB(Z%,0,TEST)
310OPT FNRMB(Z%,1,TEST)
320OPT FNRMB(Z%,2,TEST)
330OPT FNRMB(Z%,3,TEST)
340OPT FNRMB(Z%,4,TEST)
350OPT FNRMB(Z%,5,TEST)
360OPT FNRMB(Z%,6,TEST)
370OPT FNRMB(Z%,7,TEST)
380OPT FNSMB(Z%,0,TEST)
390OPT FNSMB(Z%,1,TEST)
400OPT FNSMB(Z%,2,TEST)
410OPT FNSMB(Z%,3,TEST)
420OPT FNSMB(Z%,4,TEST)
430OPT FNSMB(Z%,5,TEST)
440OPT FNSMB(Z%,6,TEST)
450OPT FNSMB(Z%,7,TEST)
460OPT FNBITI(Z%,&AA)
470OPT FNINCA(Z%)
480OPT FNDECA(Z%)
490OPT FNPHY(Z%)
500OPT FNPLY(Z%)
510OPT FNPHX(Z%)
520OPT FNPLX(Z%)
530OPT FNTSB(Z%,BGTEST)
540OPT FNTRB(Z%,BGTEST)
550OPT FNBIT(Z%,BGTEST)
560OPT FNJMP(Z%,BGTEST)
570OPT FNSTZ(Z%,BGTEST)
580OPT FNSTZX(Z%,BGTEST)
590OPT FNBBR(Z%,0,TEST,L2)
600.L2 OPT FNBBR(Z%,1,TEST,L3)
610.L3 OPT FNBBR(Z%,2,TEST,L4)
620.L4 OPT FNBBR(Z%,3,TEST,L5)
630.L5 OPT FNBBR(Z%,4,TEST,L6)
640.L6 OPT FNBBR(Z%,5,TEST,L7)
650.L7 OPT FNBBR(Z%,6,TEST,L8)
660.L8 OPT FNBBR(Z%,7,TEST,L9)
670.L9 OPT FNBBS(Z%,0,TEST,L10)
680.L10 OPT FNBBS(Z%,1,TEST,L11)
690.L11 OPT FNBBS(Z%,2,TEST,L12)
700.L12 OPT FNBBS(Z%,3,TEST,L13)
710.L13 OPT FNBBS(Z%,4,TEST,L14)
720.L14 OPT FNBBS(Z%,5,TEST,L15)
730.L15 OPT FNBBS(Z%,6,TEST,L16)
740.L16 OPT FNBBS(Z%,7,TEST,L9)
750]
760NEXT
770 REM THIS IS THE END OF
       THE TEST PROGRAM
30000REM  START OF 65C02 FUNCTIONS
30001END
30002DEFFNBRA(opt%,where%)
30003[OPT opt%:BNE where%:]
30004IFopt%>3 O%?-2=&80 ELSEP%?-2=&80
30005=opt%
30006DEFFNBBS(opt%,bit%,address%,where%)
30007[OPT opt%:BNE where%:]
30008IFopt%>3 O%?-2=&8F+bit%*16:O%?0=(O%
?-1)-1:O%?-1=address%:O%=O%+1 ELSEP%?-2=
&8F+bit%*16:P%?0=(P%?-1)-1:P%?-1=address
%
30009P%=P%+1
30010=opt%
30011DEFFNBBR(opt%,bit%,address%,where%)
30012[OPT opt%:BNE where%:]
30013IFopt%>3 O%?-2=&0F+bit%*16:O%?0=(O%
?-1)-1:O%?-1=address%:O%=O%+1 ELSEP%?-2=
&0F+bit%*16:P%?0=(P%?-1)-1:P%?-1=address
%
30014P%=P%+1
30015=opt%
30016DEFFNPHY(opt%)
30017[OPT opt%:EQUB &5A:]
30018=opt%
30019DEFFNPLY(opt%)
30020[OPT opt%:EQUB &7A:]
30021=opt%
30022DEFFNPHX(opt%)
30023[OPT opt%:EQUB &DA:]
30024=opt%
30025DEFFNPLX(opt%)
30026[OPT opt%:EQUB &FA:]
30027=opt%
30028DEFFNRMB(opt%,bit%,address%)
30029[OPT opt%:EQUB bit%*16+&07:EQUB add
ress%:]
30030=opt%
30031DEFFNSMB(opt%,bit%,address%)
30032[OPT opt%:EQUB bit%*16+&87:EQUB add
ress%:]
30033=opt%
30034DEFFNINCA(opt%)
30035[OPT opt%:EQUB &1A:]
30036=opt%
30037DEFFNDECA(opt%)
30038[OPT opt%:EQUB &3A:]
30039=opt%
30040DEFFNTSB(opt%,address%)
30041IFaddress%>&FF [OPT opt%:EQUB &0C:E
QUW address%:] ELSE[OPT opt%:EQUB &04:EQ
UB address%:]
30042=opt%
30043DEFFNTRB(opt%,address%)
30044IFaddress%>&FF [OPT opt%:EQUB &1C:E
QUW address%:] ELSE[OPT opt%:EQUB &14:EQ
UB address%:]
30045=opt%
30046DEFFNSTZ(opt%,address%)
30047IFaddress%>&FF [OPT opt%:EQUB &9C:E
QUW address%:] ELSE[OPT opt%:EQUB &64:EQ
UB address%:]
30048=opt%
30049DEFFNSTZX(opt%,address%)
30050IFaddress%>&FF [OPT opt%:EQUB &9E:E
QUW address%:] ELSE[OPT opt%:EQUB &74:EQ
UB address%:]
30051=opt%
30052DEFFNORA(opt%,address%)
30053[OPT opt%:EQUB &12:EQUB address%:]
30054=opt%
30055DEFFNAND(opt%,address%)
30056[OPT opt%:EQUB &32:EQUB address%:]
30057=opt%
30058DEFFNEOR(opt%,address%)
30059[OPT opt%:EQUB &52:EQUB address%:]
30060=opt%
30061DEFFNADC(opt%,address%)
30062[OPT opt%:EQUB &72:EQUB address%:]
30063=opt%
30064DEFFNSTA(opt%,address%)
30065[OPT opt%:EQUB &92:EQUB address%:]
30066=opt%
30067DEFFNLDA(opt%,address%)
30068[OPT opt%:EQUB &B2:EQUB address%:]
30069=opt%
30070DEFFNCMP(opt%,address%)
30071[OPT opt%:EQUB &D2:EQUB address%:]
30072=opt%
30073DEFFNSBC(opt%,address%)
30074[OPT opt%:EQUB &F2:EQUB address%:]
30075=opt%
30076DEFFNJMP(opt%,address%)
30077[OPT opt%:EQUB &7C:EQUW address%:]
30078=opt%
30079DEFFNBIT(opt%,address%)
30080IFaddress%>&FF [OPT opt%:EQUB &3C:E
QUW address%:] ELSE[OPT opt%:EQUB &34:EQ
UB address%:]
30081=opt%
30082DEFFNBITI(opt%,byte%)
30083[OPT opt%:EQUB &89:EQUB byte%:]
30084=opt%
```

Fig 2 Program listing